HO CHI MINH UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION AND TECHNOLOGY



**Course:** Introduction to Artificial Intelligence

**PROJECT 2**

**SAT APPROACH**

**FOR 8-QUEENS PROBLEM**

**Class:** 20CLC11

**Instructors:** Nguyễn Ngọc Thảo

Lê Ngọc Thành

Nguyễn Thành An

Hồ Thị Thanh Tuyến

Table of Contents

[I. Team members’ information 3](#_Toc101537087)

[II. Self-assessment of each member 3](#_Toc101537088)

[III. Check list 3](#_Toc101537089)

[IV. Task a 4](#_Toc101537090)

[V. Task b 4](#_Toc101537091)

[VI. Task e 4](#_Toc101537092)

[VII. Evidence videos 8](#_Toc101537093)

[VIII. References 8](#_Toc101537094)

1. **Team members’ information**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 20127610 | Trương Samuel |
| 20127285 | Trần Hồng Minh Phúc |
| 20127001 | Hà Quốc Anh |

1. **Self-assessment of each member**

|  |  |  |
| --- | --- | --- |
| **Student ID** | **Assigned tasks** | **Completion level** |
| 20127001 | Task a  Task f | 100% |
| 20127285 | Task b  Task e | 100% |
| 20127610 | Task c  Task d | 100% |

1. **Check list**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Criteria** | **Check list** | **Note** |
| **1** | Task a | ✓ |  |
| **2** | Task b | ✓ |  |
| **3** | Task c: Level 1 | ✓ |  |
| **4** | Task c: Level 2 | ✓ |  |
| **5** | Task d | ✓ |  |
| **6** | Task e | ✓ |  |
| **7** | Task f | ✓ | Don’t have the visualization of the state at each step of the search path and don’t create CNF text file. |
| **8** | Report + Video demo | ✓ |  |

1. **Task a**

* The input(s) and output(s) of the problem:
  + Input: size of the board (8x8)
  + Output: 8 queens are placed on the board so each pair of them doesn’t attack each other.
* The data structures that represent variables and any state of the program:
  + A 2D array(matrix) to represent the board. Cells with value 1 are where queens are placed; cells with value 0 are the valid position where queens can be placed on; and cells with value -1 are the invalid position that the queens can attack each other.
* The initial state is the empty board. The goal state is the board has 8 queens so that each pair of them doesn’t attack each other.

1. **Task b**

(Constraints for row)

b[3][0] ∨ b[3][1] ∨ b[3][2] ∨ b[3][3] ∨ b[3][4] ∨ b[3][5] ∨ b[3][6] ∨ b[3][7] ∧

-b[3][0] ∨ -b[3][3] ∧

-b[3][1] ∨ -b[3][3] ∧

-b[3][2] ∨ -b[3][3] ∧

-b[3][3] ∨ -b[3][4] ∧

-b[3][3] ∨ -b[3][5] ∧

-b[3][3] ∨ -b[3][6] ∧

-b[3][3] ∨ -b[3][7] ∧

(Constraints for column)

b[0][3] ∨ b[1][3] ∨ b[2][3] ∨ b[3][3] ∨ b[4][3] ∨ b[5][3] ∨ b[6][3] ∨ b[7][3] ∧

-b[0][3] ∨ -b[3][3] ∧

-b[1][3] ∨ -b[3][3] ∧

-b[2][3] ∨ -b[3][3] ∧

-b[3][3] ∨ -b[4][3] ∧

-b[3][3] ∨ -b[5][3] ∧

-b[3][3] ∨ -b[6][3] ∧

-b[3][3] ∨ -b[7][3] ∧

(Constraints for primary diagonal)

-b[0][0] ∨ -b[3][3] ∧

-b[1][1] ∨ -b[3][3] ∧

-b[2][2] ∨ -b[3][3] ∧

-b[3][3] ∨ -b[4][4] ∧

-b[3][3] ∨ -b[5][5] ∧

-b[3][3] ∨ -b[6][6] ∧

-b[3][3] ∨ -b[7][7] ∧

(Constraints for secondary diagonal)

-b[6][0] ∨ -b[3][3] ∧

-b[5][1] ∨ -b[3][3] ∧

-b[4][2] ∨ -b[3][3] ∧

-b[3][3] ∨ -b[2][4] ∧

-b[3][3] ∨ -b[1][5] ∧

-b[3][3] ∨ -b[0][6] ∧

1. **Task e**

* Our heuristic function will return the number of valid position that the queen can placed on.
* After placing the advance queen(s), our algorithm will find the min-row that haven't had any queen yet. Next, we will get succsesor(s) by checking which cell in that row is still able for the next queen standing. After finding all successor(s) in the row, we will add it to a queue. The queue will be sorted in ascending order basing on the heuristic value. Then take the one that have the lowest heuristic value to continue to solve the problem. The loop end when it reaches the goal state, or until the queue is empty.
* Please read the green comment code below:

Text

Description automatically generated

* This is the function of finding the successor(s). It will return the list of indexes that the queen can place on that cell.

Text

Description automatically generated

* The goal state must be required to satisfy two these conditions: the number of queens equal to the board size and none of them attacking each other.

Graphical user interface

Description automatically generated with medium confidence

* The function below is to check if our queens have attacked each other or not.

Text

Description automatically generated

* This is the sorting function, which we have mentioned above.

A screenshot of a computer

Description automatically generated with medium confidence

* If the while-loop end because of the queue is empty, we will return None to show that there is not any solutions with the advance queen(s).

Text

Description automatically generated

* For example, we will place two first queens at (2, 0) and (0, 3).

Table

Description automatically generated

* Basing on our algorithm, it will find the min-row that does not have a queen. In this case is row 1. Then, it will look for the cell that is not attacked. So, the successor will have four elements as four coordinates: (1, 4), (1, 5), (1, 6), (1, 7). Because it is required to add to the current and each element will be added to the queue. Therefore, after adding to the queue, we will have four elements:

1. [(2,0), (0,3), (1,4)]
2. [(2,0), (0,3), (1,5)]
3. [(2,0), (0,3), (1,6)]
4. [(2,0), (0,3), (1,7)]

* In the next step, we sort it base on the heuristic value and then the queue will change like this:

1. [(2,0), (0,3), (1,6)]
2. [(2,0), (0,3), (1,7)]
3. [(2,0), (0,3), (1,5)]
4. [(2,0), (0,3), (1,4)]

* This is the heuristic that we have calculated:

Text

Description automatically generated

* Now it will check the conditions of the while-loop and we see that the queue is not empty, so we continue to get the current sate. As a result, the current state is   
  [(2,0), (0,3), (1,6)] and we check if it is the goal state or not.
* Finally, continue until the while-loop end.

1. **Evidence videos**

https://drive.google.com/file/d/1jkWxyAZCC3P\_rCZn\_MGK1ZlgijMQ19JF/view?usp=sharing

1. **References**

[The N-queens Problem  |  OR-Tools  |  Google Developers](https://developers.google.com/optimization/cp/queens)

[Notes on Chapter 7: Logical Agents and Propositional Satisfiability (sfu.ca)](http://www.sfu.ca/~tjd/310summer2019/chp7_proplogic.html)

[Pygame Mouse Click and Detection - CodersLegacy](https://coderslegacy.com/python/pygame-mouse-click/)

[How to make a text input box python pygame Code Example (codegrepper.com)](https://www.codegrepper.com/code-examples/python/how+to+make+a+text+input+box+python+pygame)

[How to Draw a Chessboard in Python/Pygame #Chessboard - YouTube](https://www.youtube.com/watch?v=lle-Gtv5ANY&ab_channel=SyntaxTerminator)

[Artificial Intelligence - 8 Queens SAT solver A\* demo - YouTube](https://www.youtube.com/watch?v=6p3cul3eM1s&ab_channel=T%E1%BA%A5nPh%C3%A1tT%E1%BB%AB)